TECHNICAL INTERNSHIP

Porting NEMO ocean model to MareNostrum4

DE GIMEL Agnès
MAIN

Supervisor: Kim SERRADELL
Internship dates: 08/06/2017 to 25/08/2017

Academic Year : 2016/2017

Résumé

L'objectif de ce stage technicien était de produire une étude de scalabilité sur un modèle de simulation de la Terre sur Marenostrum IV, un supercalculateur récemment acquis par le laboratoire dont la puissance de calcul atteint 13,4 pétaflops, douze fois plus élevée que pour la troisième génération de MareNostrum. Il fallait donc déterminer le meilleur nombre de processus à utiliser en terme de performance et d'efficacité.

Le modèle comprend plusieurs sous-modèles simulant différentes parties de la Terre. Les deux principaux sont l'Integrated Forecasting System(IFS) et le Nucleous for European Modelling of the Ocean (Nemo), modèles pour l'atmosphère et pour l'océan respectivement. Ces deux modèles ont la particularité de devoir s'attendre à chaque fin de pas de temps de la simulation avant de passer au pas de temps suivant. Il faut donc chercher en priorité à réduire ce temps d'attente pour atteindre un certain équilibre et ne pas utiliser de ressources inutilement. De plus, un autre problème s'ajoute : les pas de temps d'IFS ne sont pas réguliers. En effet, en plus des calculs habituels, tous les quatre pas de temps, il doit fournir un travail supplémentaire (effectuer des calculs pour la glace), et tous les huit pas de temps, les sorties. Ainsi, il est impossible de ne pas avoir d'attente entre les deux modèles puisque même si sur un pas de temps « classique », Nemo et IFS font le même temps, il faudra ajouter ces temps supplémentaires pour ces autres pas de temps et Nemo sera contraint alors d'attendre.

Pour effectuer cette étude, j'ai eu d'abord à prendre en main des outils du laboratoire et à me familiariser au lancement d'expérience, pour ensuite automatiser des lancements avec différents nombres de processeurs pour les deux sous-modèles via un script bash que j'ai écrit. Puis, il a fallu automatiser la collection des données de sorties des expériences (temps de la simulation, temps d'attente de IFS, de Nemo etc...) via un autre script, et enfin un dernier script permettait de calculer d'autres métriques utiles et de tracer les graphiques permettant la visualisation des résultats.

Il a d'abord fallu chercher le meilleur temps de calcul pour Nemo, puis chercher à obtenir un temps similaire pour IFS. J'ai ainsi pu obtenir la meilleure combinaison en termes de performance (temps de la simulation) et la meilleure combinaison en termes d'efficacité (performance ramenée au nombre de processus utilisés)

Acknoledgements

I would especially like to thank Kim Seradell for welcoming me in BSC and giving me all the information I needed to start my internship.
Of course, I also want to thank Domingo Manubens, for helping me with handling autosubmit, and Mario Acosta, for explaining me how the models work and how to use Lucia.
I also want to thank all the people of BSC : researchers, PHD students and interns who welcomed me and helped me during my internship.

Contents

Introduction

The Barcelona Supercomputing Center (BSC) is a public research center located in Barcelona, Catalonia, Spain. It hosts MareNostrum, a 13.7 Petaflops, Intel Xeon Platinum-based supercomputer, which also includes clusters of emerging technologies. With this computer, a lot of studies are conducted in many different areas : astronomy, earth sciences, space sciences, biomedecine, physics and engineering, computer sciences, chemistry or material sciences for example. As for me, I was in the department of Eart-Science.

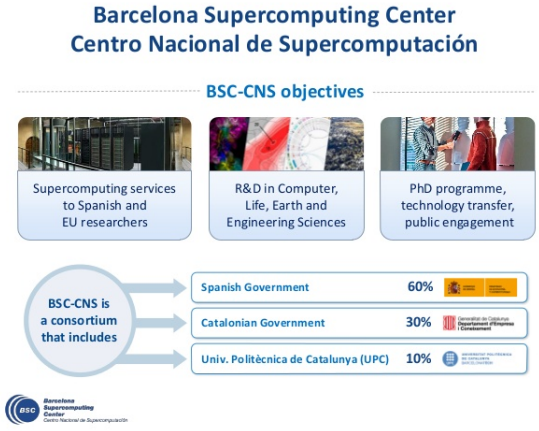

Figure 1: Bsc objectives

During this technical internship, I had to do a scalability study of an Earth System model known as EC-Earth, on a new version of the supercomputer : Marenostrum IV. This model was parallelized using MPI, a distributed memory paradigm. This means that its speed can vary depending on how many processes are used by the model. For the execution, It is important to achieve a good parallel efficiency in order to reduce the energy needed for the simulations and obtain the results faster. Knowing the scalability of the model is trivial because adding processes to the model will not necessarily accelerate the model, especially if we already reached the maximum speedup.

Therefore, it is important to find the best combination. The main work of my intership was to automatise via scripts (written mainly in bash, but also in python) the launching of experiments with different number of MPI processes (defined in the script), then to collect the information once the experiments are over, and finally to plot the graph to visualise the scalability of the model.

The main difficulty was that the model is composed by two main different components : the Nucleous for European Modelling of the Ocean (NEMO) that is simulating the ocean, and one simulating the atmosphere : the Integrated Forecasting System (IFS). This represent a Multiple Program, Multiple Data (MPMD) application in parallel. Therefore, I had to find the best combination of processes for both components.

I also had to take into account the fact that the two components are not completly independants. The simulation is divided into time-steps using a finite difference approach for the model. At the

end of each time step, both components have to exchange some information as boundary conditions, this means that the fastest model has to wait for the other one to finish before moving to the next time-step. Therefore, improving the execution time of one component didn't mean improving the speed of the global final time.

The purpose was that, at the end of my internship, to know better how to use the resources (processes) of Marenostrum to run those simulations, in terms of performance (speed of the simulation) and efficiency (using the minimum processes needed), in order not to "waste" energy.

1 Ec Earth model and its simulations

1.1 Ec-Earth components

EC-Earth is a global coupled climate model, which integrates a number of components models in order to simulate the earth system. It is developped by a consortium of European research institutions, which collaborate in the development of a new Earth System Model.
The goal of Ec-Earth is to build a fully coupled atmosphere-ocean-land-biosphere model usable for problems encompassing from seasonal-to-decadal climate prediction to climate change projections and paleoclimate simulations. It includes the following components.

1.1.1 L´OASIS3-MCT coupler

It is a coupling library to be linked to a component models and which main function is to interpolate and exchange the coupling fields between them to form a coupled system.

1.1.2 Integrated Forecating System (IFS)

It is the atmoshperic model, an operational global meterological forecasting model developed and maintained by the European Centre of Medium-Range Weather Forecasts (ECMWF).
The dynamical core of IFS is hydrostatic, two-time-level, semi-implicit, semi-Lagrangian and applies spectral transforms between grid-point space and spectral space. In the vertical the model is discretised using a finite-element scheme. A reduced Gaussian grid is used in the horizontal. The IFS cycle is 36r4.

1.1.3 Nucleous for European Modelling of the Ocean (NEMO)

It is the ocean model, a state-of-the-art modelling framework for oceaonagraphic research, operational oceanography seasonal forecast and climate studies.
It discretizes the 3D Navier-Stokes equations, being a finite difference, hydrostatic primituve equation model, with a free sea surface and a non-linear equation of state in the Jackett. The ocean general circulation model (OGCM) is OPA (Océan Parallélisé). OPA is a primitive equation model which is numerically solved in a global ocean curvilinear grid known is ORCA.

Ec-Earth 3.2.0 uses NEMO's version 3.6 with XML Input Output Server (XIOS) version 1.0. XIOS is an asynchronous input/output server used to minimize previous I/O problems.

1.1.4 The Louvain-la-Neuve sea-Ice Model 2/3 (LIM2/3)

It is a thermodynamic-dynamic sea-ice model directly coupled with OPA.

1.1.5 The Hydrological extension of the Tiled ECMWF Surface Scheme for Exchange processes over Land (HTESSEL)

It is the land and vegetation module (is part of the atmosphere model).
It solves the surface energy and water balance taking into account 6 different land tiles overlying a 4 layer soil scheme. The 6 tiles are : tall vegetation, low vegetation, interception reservoir, bare soil, snow on low vegetation, and snow under high vegetation.

1.1.6 The Tracer Model 5 (TM5)

It is a global chemistry transport model : it describes the atmospheric chemistry and transport of reactive or inert tracers.

1.1.7 The runoff-mapper

This component is used to distribute the runoff from land to the ocean through rivers. It runs using its own binary and coupled through OASIS3-MCT.

Here is a scheme of the different components and how they are connected.

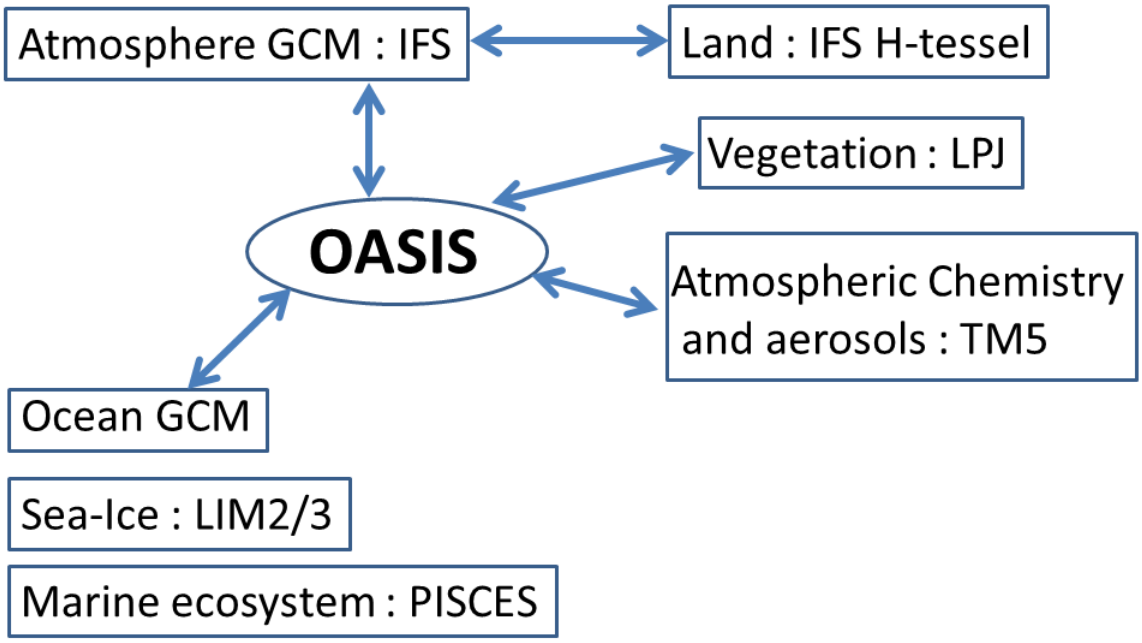

Figure 2: Components of the Ec-Earth model

1.2 Coupling between components

IFS and NEMO are components of EC-Earth that interact between them at the end of a time step (in general, one simulated month) to harmonize the calculated values.
Indeed, their grids are not independant and intersect : both components have to communicate at the end of a time step to share values and exchange information that they will need for the next time step, this process is known as coupling.
Thus, the more frequent are those communications (short time steps), the more precise is the model, but the overhead produced by MPI communications increase.

Therefore, the components are independant unti the end of each calculation time : they first start to calculate their equations (for ocean and atmosphear) alone, but once it is over, they have to communicate boundary conditions and update their values for these boundaries, and finally be

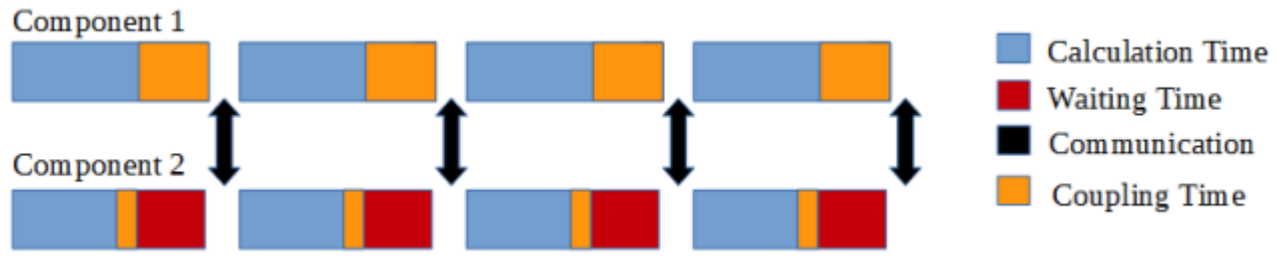

Figure 3: Execution of 4 time steps for the two components.

able to start the next time step simulation.
This "dependance" makes the scalability tests more difficult.
Indeed, by making one of the component go faster does not mean make the simulation goes faster, this is because it may have to wait for the other to finish at every time step.

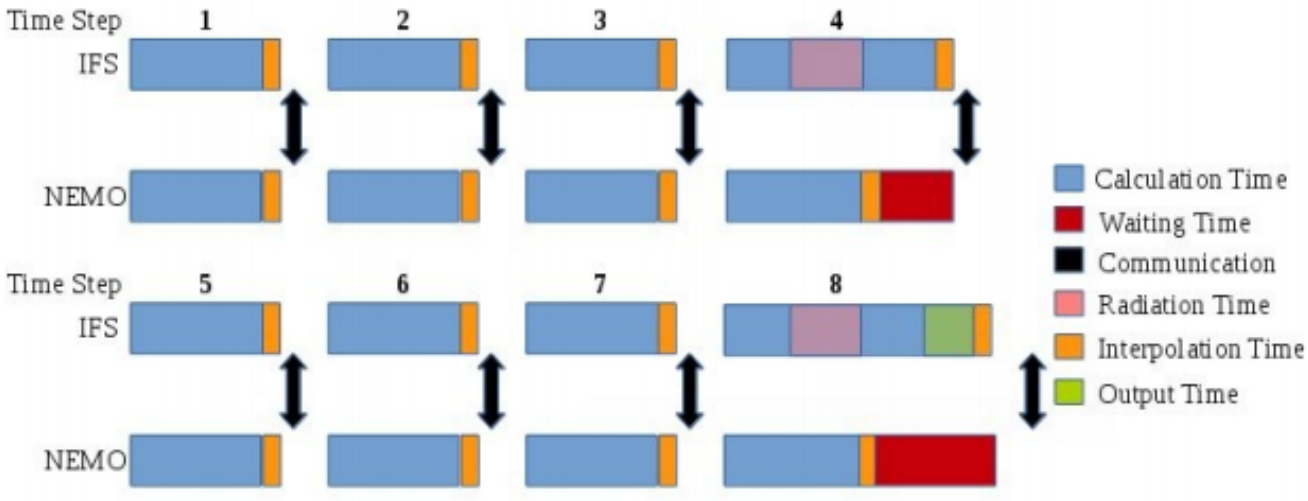

Figure 4: 8 time steps of the coupling model

Additionally, the calculation time of IFS is not regular. Indeed, every 4 time steps, it also measures the "radiation time" and every 8 time steps it has to do the output. Thus, contrary to NEMO, for which the total time of execution represents only the calculation time, the interpolation time and the waiting time, the calculation time of IFS also includes the radiation time and the output.

2 Scalability on Marenostrum IV

2.1 The decomposition into different scripts

The purpose of the internship was to provide a scalability study of the model on the platform marenostrum IV, a new supercomputer of the BSC.

The main difficulty of this work was that two parameters had to be modified : the number of processes of IFS and the number of processes for NEMO.

The most time-consuming part of this work was to automatise the launch of the experiments, thanks to scripts in bash, and then to collect the output to calculate other metrics and create the plots with the results to analyse those metrics and how the number of processes of IFS and NEMO influence them. To do so, I had to write 3 scripts.

Here is a simplified scheme showing how the scripts were organised. Lucia is the tool used to calculate metrics thanks to the completed output files of the experiments. SYPD is a metric that will be explained further in the report.

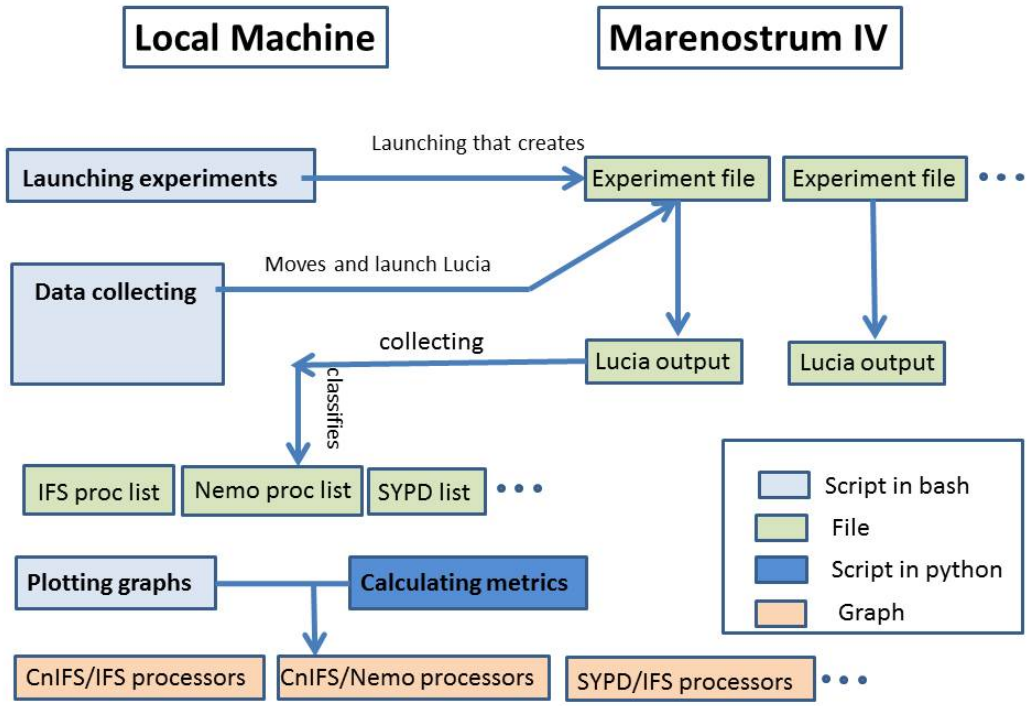

Figure 5: Organisation of the different scripts

The first one was written to launch the experiments with different numbers of MPI processes for IFS and for NEMO.

For this, I took another script already used for a similar job on another platform. I modified the parameters and also added two loops. I needed two independant loops : the first one was to modify the number of processes of IFS with a fixed number of processes of NEMO, and the second one was to modify the number of processes of NEMO with a fixed number of processes for IFS.

The second script was to automatise the collection of the output of the different experiments. To do so, I was given a tool (an executiable file) that could print the output needed if I launched it in the directory of the experiment.

The purpose was here to automatise the launch of this tool in the files of the different experiments. However, other infomartion had to be taken from other output files.
Those output collected were then socked in files. I had to make the difference between the output of the experiments of the first loop (modification of the number of processes of IFS) and the experiements that resulted from the second loop (modification of the number of processes of NEMO) because they represent different metrics.

Lastly, the completion of one last script was needed to plot the results using graphs that would allow to represent the scalability study. Other additional complex metrics are calculated here too, thanks to another code written in python, that could deal with double.

2.2 The metrics

To complete the scalability study, it was important to understand the different metrics. Some of them were directly available in the output files or through the executable file that could plot them while others had to be calculated thanks to the previous ones.

The metrics that were directly collected were the following :

- execution time of the model
- CNNEMO :total calculation time of NEMO.
- CNIFS : total calculation time of NEMO : this time also includes the "radiation time" and the "output time".
- WAITINGIFS : total waiting time of IFS
- WAITINGNEMO : total waiting time of NEMO
- Simulated Years Per Day (SYPD).
  It is the number of years simulated in one day. This metric is calculated as follows :

$$SYPD = (\frac{3month}{executiontimes})(\frac{1year}{12month})(\frac{3600s}{1h})(\frac{24h}{1day}) = \frac{21600}{Executiontime(s)}Years/Day \quad (1)$$

- Total time-step (Tts) : number of steps.

The metrics calculated a posteriori are the following :

- MinTotal

$$MinTotal = 1.3 * MIN(RTS\_IFS, RTS\_NEMO) * Tts/8 \tag{2}$$

- RTS_NEMO, is the calculation time of NEMO on one time step. It is calculated as follows :

$$RTS\_NEMO = \frac{CnNEMO}{Tts} \tag{3}$$

- RTS_IFS, is the calculation time of NEMO on one time step. It does not include the "radiation time" and the "output time". It is important to mention that 9,3 was an approximative value of the extra time on 8 time steps, taking prom previous simulations.

$$RTS\_IFS = \frac{Cn\_IFS * 8}{Tts * 9,3} \tag{4}$$

- unbalance percentage, it measures the imbalance between RTS_IFS and RTS_NENO and is calculated as follows :

$$PercUnbalance = \frac{RTS\_IFS * 100}{RTS\_NEMO} - 100 \tag{5}$$

- IdealCNIFS, represents the "best" CnIFS that could be obtained if RTS_IFS and RTS_NEMO were balanced. It is calculated as followed :

$$IdealCnIFS = \frac{Min(RTS\_IFS, RTS\_NEMO) * 9.3 * Tts}{8} \tag{6}$$

  The 9.3 was founded empirically, we estimated the difference of time between a classical time-step and a time step that included the radiation time and the output time.

- IdealCNNEMO, same as IdealCNIFS.

$$IdealCnNEMO = Min(RTS\_IFS, RTS\_NEMO) * Tts \tag{7}$$

3 The results

3.1 First tests and visualisations of correlations of the metrics

The first tests produced the following graphs. Here is the total calculation time of both models (CnIFS ad CnNemo) based on the number of processes used.

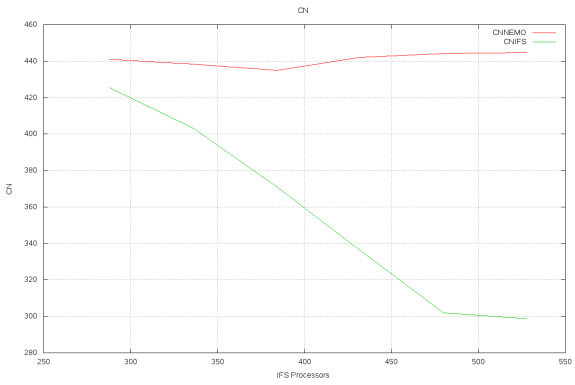

Figure 6: Total calculation time of Nemo and IFS with Nemoproc=96

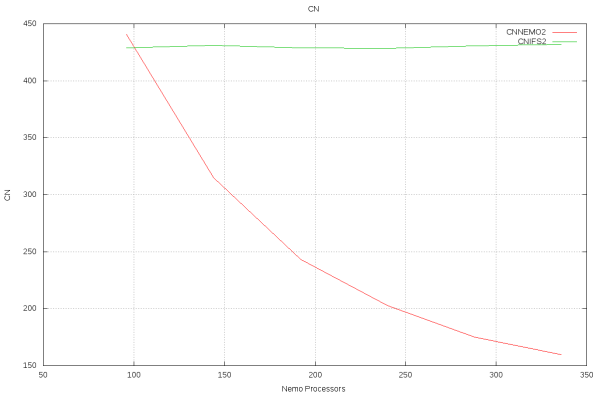

Figure 7: Total calculation time of Nemo and IFS with IFSproc=288

Firstly, we can notice that with enough processes (more than one hundred for Nemo with IFSproc=288), Nemo is faster than IFS. This result was expected, due to the fact that IFS is not regular and has a heavier workload every 4 time-steps. The best way to visualise this gap is thanks to the unbalance percentage.

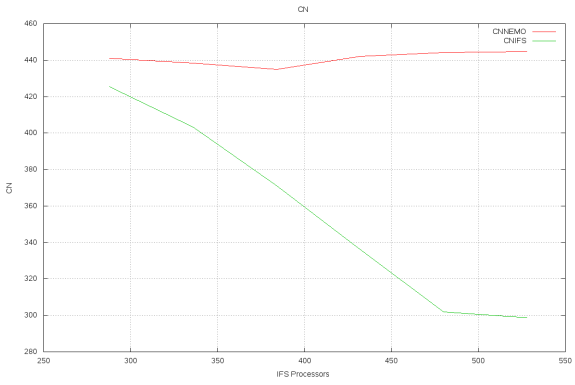

Figure 8: Total calculation time of Nemo and IFS with Nemoproc=96

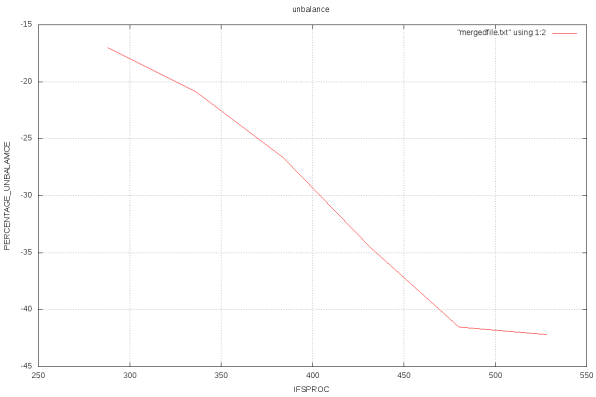

Figure 9: Percentage of unbalance with Nemoproc=96

We can see the correlation of the two graphs : the wider the gap between the calculation times of the two components(graph on the left), the wider is the percentage of unbalance between them (graph on the right, the values on the ordinate axis are negative). This means that the faster component (in this case NEMO) must wait before starting the next time-step.

Thus, increase the unload balance between both components is not interesting since the final execution time of EC-Earth will not be reduced and some resources will be wasted.

Therefore, the imbalance translates the gap between the speed of the two components. This result can also be found on the following graphs :

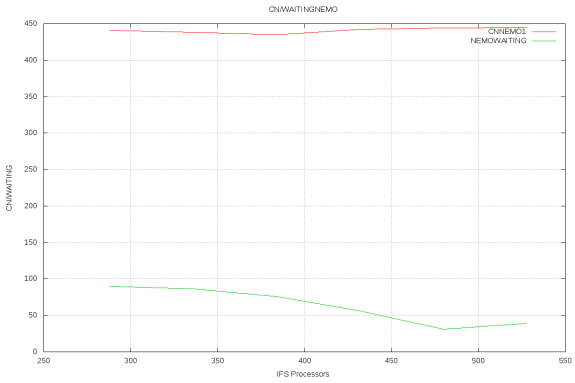

Figure 10: Waiting time for Nemo with Nemoproc=96

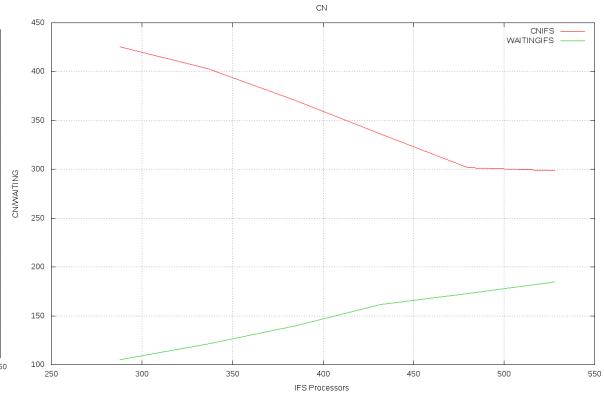

Figure 11: Waiting time for IFS with Nemoproc=96

The main information that we can see (especially the one on the right) is the correlation between the waiting time and the calculation time. By adding new processes for IFS, this component will calculate faster but it will be forced to wait more for Nemo (at least for the time-steps where it doesn't have extra calculation to do).

Lastly, the graph that is perhaps the most representative of the performance is the one showing the number of simulated years per day.

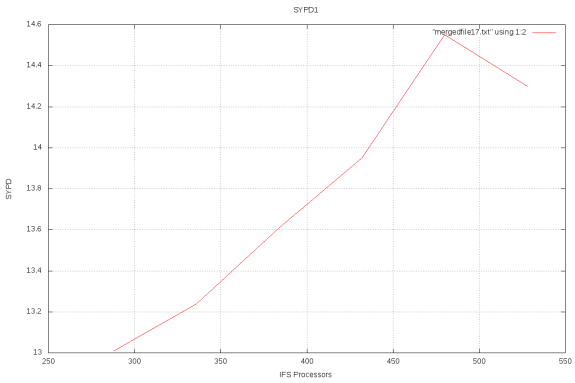

Figure 12: SYPD Nemoproc=96

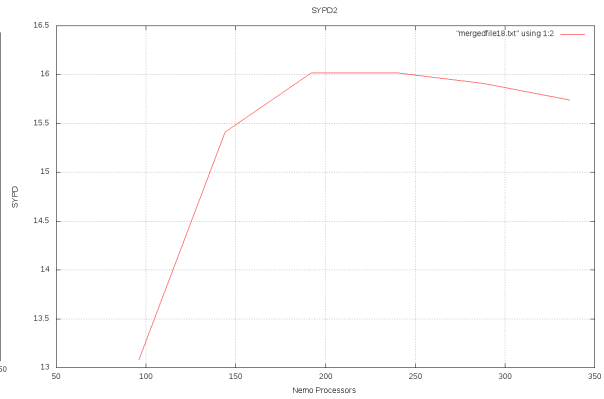

Figure 13: SYPD IFSproc=288

We see here on the left graph that the number of simulated years per day peaks when the number of processes for IFS is 480, and on the graph on the right that it peaks when the number of processes of Nemo is 192.

This result was already noticeable on the graph which shows the calculation times, for which the calculation times of IFS (being the slowest model) was the shortest. The metrics are strongly correlated and the different graphs are quite redundant. Thus, the graph representing the calculation times and the SYPD are enough to analyse the performance and the efficiency of the simulation for the next tests.

Here the objective was to accelerate as much as possible EC-Earth by using as few processes as possible for the two main components of the model. To do so, the first objective was to find the best calculation time (this is the minimal execution time to finish the simulation in parallel) for IFS (which is the slowest model from previous tests). The second stage is to find it for NEMO, taking into account previous result obtained for IFS. Indeed, this methodology can be also used to reduce the percentage of unbalance between both components, this is the waiting time (especially the one of Nemo), which will also help to increase the efficiency, reducing the waste of resources.

3.2 Finding the best combination

3.2.1 Best number of processes for IFS

The approach here was to fix the number of processes of Nemo to find the best number of processes of IFS (the one giving the shortest execution time). The number for Nemo was fixed to 480. It was chosen low enough to use as many processes for IFS as possible, because it is the model that needs to be accelerated. We get the following graphs :

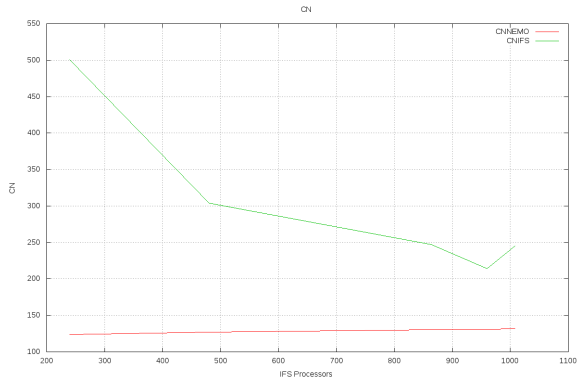

Figure 14: Calculation time of IFS and NEMO with Nemoproc=480

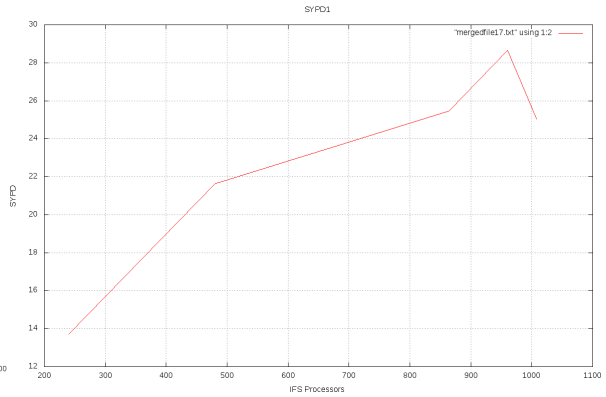

Figure 15: Simulated Years per Day Nemproc=480

We see here that the best number of processes for IFS is 960 : for this value the calculation time is the shortest and the simulated years per day is the greatest. This value will be fixed to find the best number of processes for Nemo and repeat the process.

3.2.2 Best number of processes for Nemo

The number of processes for IFS was fixed to 960 as found before. We finally get the following graphs :

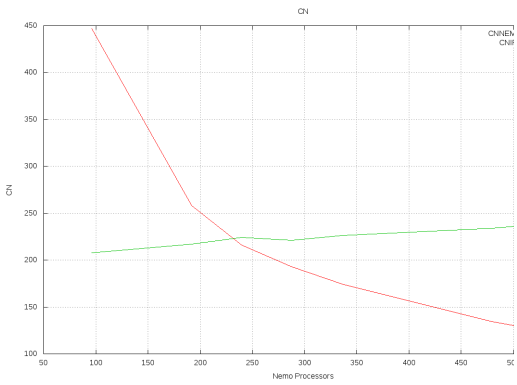

Figure 16: Calculation time for IFS and Nemo with IFSproc=960

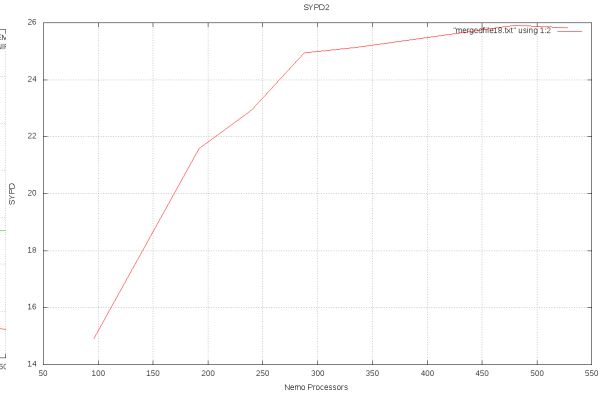

Figure 17: Simulated Years per Day IFSproc=960

We notice here that the calculation time for Nemo becomes efficient starting from 240 processes (below that, it is the slowest model and it is the one that slow down the simulation so IFS has to wait.

From this value, using more processes for Nemo accelerates the simulation slightly, it continues to acccelerate until 480 processes. After this value, the speedup of nemo is equal or less to the value achieves with 480.

If we take a look at the graph on the right, we also notice that the best we can get is 26 simulated years per day using 480 processes for Nemo.
However, with 288 processes, we get 25 simulated years per day. That means that when we use 60% of the processes used previously, we decrease the simulation year per day only a 4%

As a conclusion, we get two possible optimal combinations : the first one is using 960 processes for IFS and 480 for Nemo, we then get the best performance. However, in terms of efficiency, it is more interesting to use 960 processes for IFS and only 288 for Nemo.

Conclusion

During this internship, my mission was to provide a scalability analysis of the earth coupled model known as EC-Earth, and see what was the best combination of processes in terms of performance (speed of the simulation), or efficiency (performance and the "cost" of the simulation, which means the number of processes used in the process).

The main difficulty was that, being a coupled model, I had to do graphs for both IFS and Nemo and analyse the efficiency of both of them independently, taking into account the MPMD approach of the execution. I also had to think on how to proceed to find the best combination. I finally looked for the best performance for IFS and Nemo independantly, since the components are not dependant on the calculation times. The methodology that I followed found optimal was 1) Obtain the best calculation time for IFS and take the number of IFS processes, 2) Obtain the best calculation time of Nemo with that number of processes for IFS.

The most time-consuming part was to automatize the launching of experiments with different number of processes, and then to collect the output to visualise how the number of processes for both components impact in the execution time of the simulation.

During this internship, I wrote 4 full scripts which automatize all the process and produce the results. The first one in bash for the launching of the experiments with differents numbers of processes for each one. The second one (in bash as well) was to collect the data. The two last ones (one in bash and one in python) was to calculate other usefull metrics and to build the graphs to visualise the results of the performance.

As a result from the scalability study using the scripts, I found two possible combinations for the new platform (Marenostrum4) and EC-Earth executions. For the performance, I found that 960 processes for IFS and 480 for Nemo was the best combination. However, for the efficiency, it is better to use 960 processes for IFS and only 288 for Nemo, in order to not waste resources.

References

[1] 20 July 2016. Scalability and Performance analysis of Ec-Earth 3.2.0 using a new metric approach (Part I).

[2] 2017. PERFORMANCE ANALYSIS OF EC-EARTH3.2: LOAD BALANCE.

[3] 2017. PERFORMANCE ANALYSIS OF EC-EARTH3.2: COUPLING.